REPORT R-390 AUGUST, 1968

COORDINATED SCIENCE LABORATORY

AN ASSOCIATION PROCESSOR FOR INFORMATION RETRIEVAL

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

This work was supported in part by the Joint Services Flectronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under contract DAAB-07-67-C-0199; and in part by OE C-1-7-071213-4557.

Acknowledgment

I wish to express my gratitude to Professor R. T. Chien for his guidance, invaluable advice and suggestions throughout the period of research for this report. Thanks are extended to Mrs. Julie Anson for her preparation of this manuscript.

TABLE OF CONTENTS

Chapter 1

Introduction

The area of information retrieval has recently attracted an increasing amount of attention due to the constantly growing store of knowledge in today's world. Problems in this area involve the storage and retrieval of very large amounts of information on which very little, if any, processing is required.

A fundamental task in this area is the location of a particular set of data items which fulfills certain search requirements. This is an associative process. Data are located by specifying part of their contents and not by using the address of a storage location.

Contemporary computing systems, of the von Neumann type, cannot handle problems of this nature efficiently. This is due to the fact that all stored information is located by an address which has no relation to the content of the data stored. In these processors an information retrieval problem would require either a direct search, location by location, of a large part of the data store, or the formation of directories which would limit the serial search. The first method is unattractive because of the large search time involved for a large data base. The second method allows a somewhat shorter search time but is undesirable for other reasons. Directories must be stored in part of the memory which could otherwise be utilized for storing the data base. Also, the directories must be periodically updated as new information is added to the memory.

A processor in which the basic storage medium consists of an associative, or content addressable, memory with distributed logic would greatly reduce the problems involved in information retrieval. In such a system data would be addressed directly by specifying a known part

The object of this report is to describe an information retrieval system which operates in a highly parallel fashion in an associative mode. By associative it is meant that a particular item is located in the data base by content addressing. In this process an entire record of information is located using a known part of the record as the search criteria. As an elementary example, the record "John's car is bright blue" could be located or retrieved by specifying the words "blue" and/or "car." This method of data interrogation is desirable in all information retrieval problems. The notion of addressing by contents enables us to avoid the artificialities of location assignment and frees us to a large extent from such local considerations as scanning, searching and counting. The operation is parallel due to the fact that the entire contents of the memory is searched simultaneously. This is accomplished by distributing a sufficient amount of logic throughout the memory. This allows each storage location or "cell" to act, to a certain degree, as a small independent processing unit.

A number of papers on the subject of associative memory processors with distributed logic have already appeared. Designs for such processors have been proposed by Lee and Paull[1], Lee[2], Gaines and Lee[3], Sturman[4], and Hayes[6].

The associative memory processor described below is a logical outgrowth of the organization proposed by Lee and Paull[1]. The previous organizations were thoroughly investigated and it was found that they were

somewhat deficient in two areas. First of all, the general operation of the processors was found to be inefficient in terms of the number of steps (therefore the amount of time) and the amount of programming required to perform a data search. Secondly, but most important, it was found that certain types of searches which are extremely important to information retrieval could not be performed at all.

Chapter 2 outlines the general operation of previous distributed logic associative memory processors by briefly describing the organization proposed by Lee and Paull[1]. The shortcomings of this system and qualities which are desirable in a processor of this type are also presented.

In Chapter 3 the structure of a new associative processor which alleviates these problems is described.

In Chapter 4 the data and instruction formats of this system are described.

Finally in Chapter 5 a number of examples and programs are given which illustrate the application of this system to problems of information retrieval.

Chapter 2

Previous Processors and Some Problems

2.1 The Processor of Lee and Paull

The machine proposed by Lee and Paull consists of an associative memory connected to a conventional stored program computer. The primary function of the computer is to act as a control device for the associative memory. It also must be able to store and execute the search programs and temporarily store data which is retrieved from the memory. The data base which is to be interrogated is stored in the associative memory.

The basic memory is a linear array of small identical sequential state machines called cells. Each cell contains both a number of binary storage elements and a sufficient amount of logic to enable it to perform the functions of the cell. The storage elements of the cells are of two types: the symbol elements and the activity elements. The symbol elements are those in which the information or "symbol" of the cells are stored. The activity elements are used as bookkeeping tags to keep track of particular cells during the operation of the machine. Each cell in the array is connected to a set of common bus lines, which include the input leads, the output leads and the various control leads which provide the cells with commands. A cell C_i also has the ability to transfer its activity status to either of its immediate neighbors, C_{i+1} or C_{i-1}. The contents of every cell (the symbol and/or the activity) can be matched against the pattern presented by the control computer on the input bus lines. In this way all cells containing a particular bit pattern can be

located and tagged.

Information is stored in the computer as strings of symbols with each symbol stored in a different cell. Strings can be of any length as long as there is room to store them. Consecutive symbols of a string are stored in consecutive cells. Therefore if a string contains n symbols and the first symbol of the string is stored in cell C_i, then the second symbol is stored in C_{i+1}, the third in C_{i+2}, and so on with the nth symbol being stored in the C_{i+n} cell. The location of a string is completely arbitrary since the cells do not have addresses, but the order of the cells in which consecutive symbols are stored is very important. A special symbol α_o is provided for the beginning of a string. Parts of strings or parameters are distinguished by storing another special symbol β after them. Therefore if a string consisted of the parameters XY, OW, PHD, it would be stored in the cell memory in the following manner: α_oXYβOWβPHDβ, with α_o stored in cell C_i, X in C_{i+1}, Y in C_{i+2} and so on.

Identification and retrieval of a string or a particular set of strings is accomplished by using a particular parameter as the search criteria. For instance, if it were desired to locate all strings which contained the parameter OW the search would proceed in the following way. First, every cell in the memory would match its contents against the input bus lines which would contain the pattern β. This denotes the beginning of the parameter βOWβ. If a cell has a successful match it will set an activity bit. At this point all cells containing β have an activity bit set. Next, all cells with activity bits set are told to propagate this activity to their neighbor with a higher index (to the right) and then reset their own activity. Now all cells compare their contents with the input pattern consisting of a symbol equal to 0 and a set activity bit.

Only those cells which contain a set activity and the symbol 0 are allowed to keep their activity set. All other previously active cells are reset. At this point only those cells which are at the beginning of parameters and contain the symbol 0 are active. Again the activities are propagated to the right and search is made for all active cells containing the symbol W, with all other activities being reset. Now only those cells containing W which is the second symbol of all parameters having 0 as the first symbol are active. Finally the activity is again propagated to the right and a search is made for all active cells containing the symbol β. At this point only those strings which contain the parameter OW have an active cell. This cell is the cell containing β which follows OW.

This searching process is a very efficient one. It can be thought of as eliminating useless information rather than a searching process for useful information. Although the search is done serial by character it is a parallel operation conducted in each of the strings simultaneously.

Although at this stage all the strings containing the parameter OW have been isolated, more processing is required before these strings can be retrieved. First, a priority system must determine which of the strings of the set is to be retrieved first. After this string is found the activity present in this string must be transferred to the cell containing the α_o, head cell, of that string. This is accomplished by propagating the activity to the neighboring cell with lower index (the cell to the left) and matching for a cell which is active and contains the symbol α_o. If the match is successful the head cell of that string has been found. If it is unsuccessful the process is repeated until the head cell is found. From this point the characters in the string are read out

sequentially one at a time by successively propagating the activity to the right neighbor and ordering the active cell to place its contents on the output lines. This process of course is non-destructive.

This example introduces two of the shortcomings of this processor. First, the priority system of this machine only allows one string of the set to retain an active cell. This string is the one to be retrieved. Since they no longer contain an active cell the remaining strings in the set must again be identified by another search in order to be again eligible for retrieval. Provision is made to eliminate previously retrieved strings from subsequent searches, but still the same search operation must be performed as many time as there are strings in the set to be retrieved. These repetitive searches require additional operating time during which no really useful processing is being accomplished. It would be desirable to be able to retain the active status of all the strings of a set during the retrieval of the entire set. This would require that only one search, the initial search, be made. The second deficiency to be noted here is the waste of processing time needed to find the head cell in a string chosen for retrieval. The activity must be propagated to the left and a match be performed as many times as the number of cells between the head cell of the string and the active cell which resulted from the search. If this number is large the time involved to complete this process becomes very costly because the rest of the memory must stand idle until its completion. This suggests that it would be advantageous if all the cells in a string could communicate directly with the head cell without the necessity of involving other cells.

2.2 Other Problems

Additional areas in which this processor and all other similar associative memory processors [2,3,4] designed for information retrieval are deficient can be placed in two general categories. The first contains those operations the machine can execute but does not do so efficiently. The two mentioned above fall in this class.

Another problem would be that of identifying the set of all strings which contain two or more particular parameters. If the parameters are stored in a particular order and they are stored in contiguous groups of cells the searching process is the same as above, e.g., A and C are the parameters which the strings must contain, and they are stored in all strings of the set containing A and C in the form $\beta A \beta C \beta$. If one or both of these requirements is not met (as is usually the case) the searching process becomes quite complicated and requires a great amount of processing time.

Still another problem which has not been handled efficiently is that of repacking the strings in the memory after information has been eliminated. This is necessary so that all the empty cells in the memory can be utilized to store new strings of arbitrary lengths. The methods of gap closing proposed by the various authors involve programs which are extremely inefficient with regard to the amount of processor time it takes to repack the memory.

The second category includes those types of operations which are necessary for an efficient information retrieval system [5] but cannot be performed by the type of associative processor suggested by Lee and Paull and its descendents [3,4].

The organization described above identifies the set of all strings which contain the parameters of the search criteria exactly. It cannot perform a threshold search. A threshold search can be described as the identification of all strings which contain at least a certain number of the parameters used for the search. For example, it might be desirable to locate all strings which contain any three or more of five particular parameters.

A second type of search which would be useful would be a weighted threshold search. In this identification procedure the presence of a particular parameter in a string may be of more or less value than that of other parameters. In this operation each parameter used in the search would be given a weight or value corresponding to its importance. If a string contains a particular parameter this value is recorded and added to the results of previous and subsequent parameter searches. After all parameters have been searched only those strings which contained a total search value greater than a certain amount would be of interest for retrieval. Both types of searches described above imply that the strings should be capable of storing the results of many searches. No provision is made for this in the associative memory processor described above.

The following chapter is a description of the structure of an associative memory processor which will alleviate the problems outlined above and will facilitate a more efficient information retrieval system.

Chapter 3

Processor Description

The macro-structure of the processor described below closely resembles the machine described by Lee and Paull. This structure is shown in Figure 1. Again, the associative memory consists of a one-dimensional array of small identical sequential state machines called "cells." Each of these cells is connected to a common set of bus lines which transfer information symbols and control commands to and from the cells. These lines originate in the stored program digital computer which supervises the operation of the entire processor. This computer must have the capacity to store and execute the various search programs required and temporarily store the information which is transferred into and out of the associative memory.

The basic differences lie in the construction of the memory cell, and the various modes of local communication each cell enjoys with its neighbors these differences allow processing to take place on three distinct levels in the machine simultaneously. This assures a high degree of parallel operation.

In the first level each basic storage location or cell has the ability to manipulate the data stored within it with respect to the signals present on the common bus lines, but independent of what is taking place in other cells. The second level allows a contiguous group or a string of cells to function together as a separate unit within the memory. This permits a more sophisticated form of processing in which different

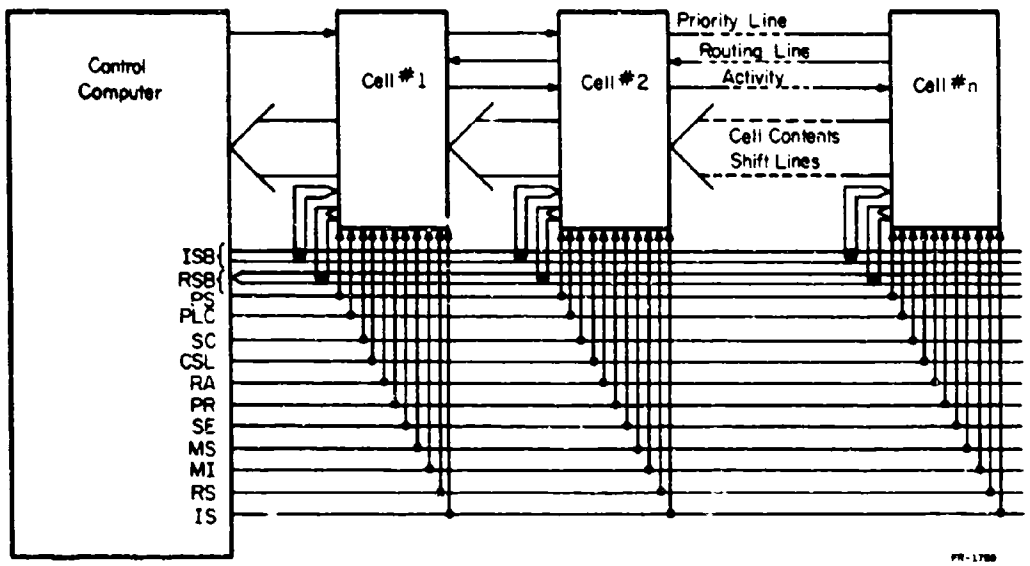

Figure 1

Macro-structure of associative processor

operations necessary to the string are performed in distinct cells. The third level is achieved by viewing the entire processor as a single unit. The program executed in the control computer directs the simultaneous operation of a large number of cells in many strings of the memory.

3.1 The Cell

All cells in the associative memory are identically the same. The logical structure of one of these cells is shown in Figure 2. Each cell contains a number of binary storage elements (bits) and a sufficient amount of logical hardware to perform the various functions required.

The binary storage elements of each cell are divided into four different fields. The first is an n+1 bit register which stores the character or symbol of the cell. This is the information register. The remaining three fields are of one bit each and are used to perform various functions which are vital to the processor's operation. The X bit is the activity of the cell. This bit is used as a bookkeeping tag to identify particular cells during processing. The Ø bit is used to indicate if a cell is empty or contains information which is no longer considered important. This bit also plays an important role in both determining cell priority and repacking the memory. The last field is the α register. The presence of this bit allows a string of cells a high degree of processing power.

3.1.1 Cell States

A cell may function in one of two states. The state of a particular cell is determined by the condition of its α binary. If this bit is

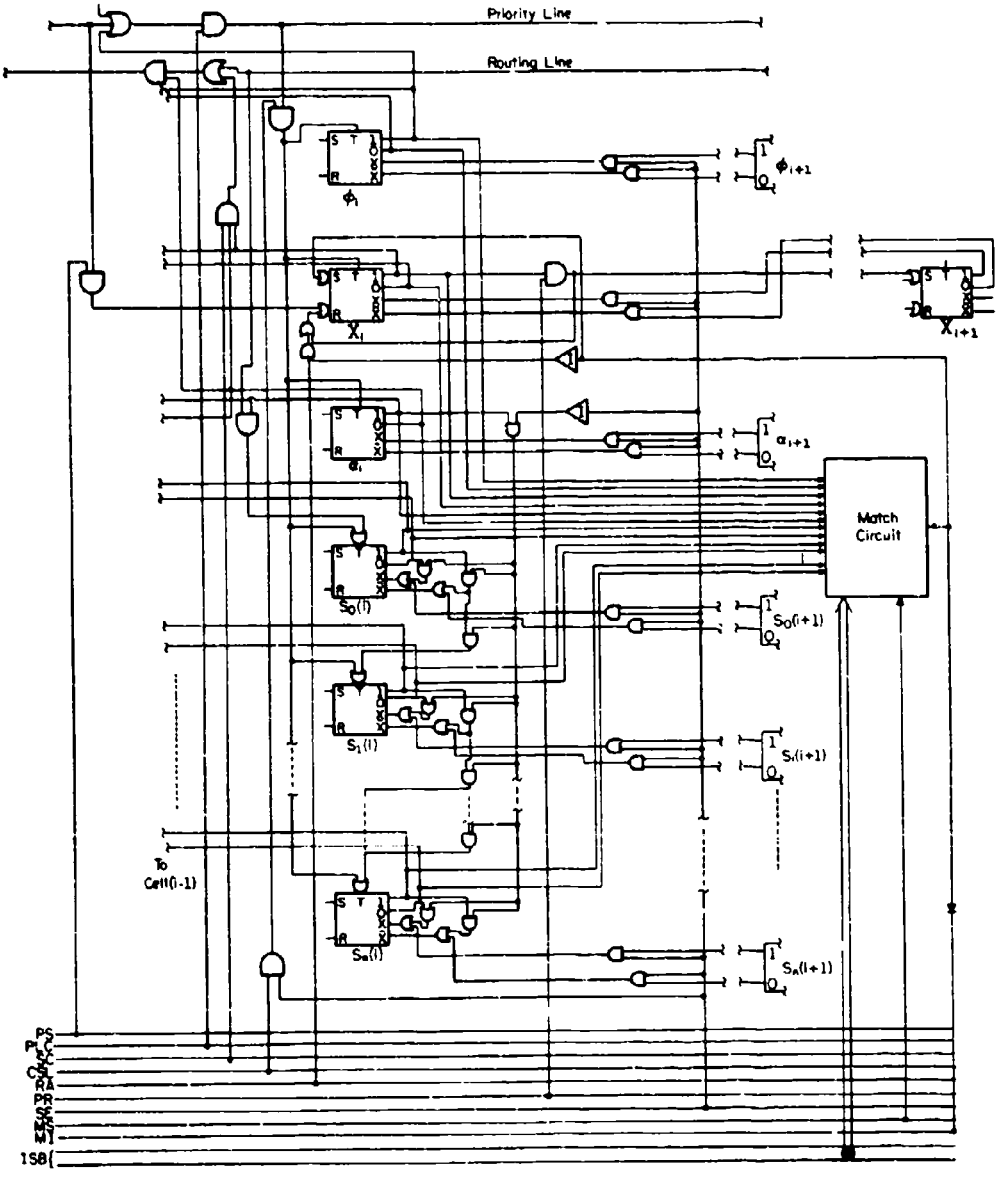

Figure 2

Structure of cell i with

input and output circuitry omitted

reset the cell operates in its normal state as a symbol storage register. If the α binary is set the cell acts as the head cell of a string. In this state the cell is able to receive count pulses which can be emitted from all the cells to the right (of higher index) up to the next head cell. The binary representation of the number of these pulses are stored in the head cell's symbol register which now acts as an accumulator. These pulses are delivered to the head cell via a special gating network called the routing line. The operation of this line is explained more fully in Section 3.3.

3.1.2 Intercell Communication

As mentioned before each cell in the array maintains direct communication with the control unit via the common bus lines. These include the input symbol bus lines, the read symbol bus lines and the various command lines. By means of the input symbol bus lines (ISB) the control presents a pattern to the array which can be used to match against the contents of each cell or which can be directly stored into the register of a selected set of cells. The read symbol bus lines (RSB) provide a path for transmitting the contents of a particular cell to the control computer. The various command lines are used to control the operation within the array.

Aside from this common mode of communication each cell also has the ability to communicate with other cells in the array in four distinct ways. First, a cell is capable of transmitting the set condition of its activity bit to its immediate neighbor of higher index (to the right). Secondly, as mentioned above, an active cell in a string can transmit a

pulse directly to the head cell of the string via the routing line. In the third case, each cell has the ability to shift the contents of all of its registers to its neighbor with lower index (to the left). This is used when repacking the memory and can also be used to load or unload the entire contents of the memory. Last of all, the cells communicate along an additional gated network called the "priority line." This network passes through all the cells in the memory and serves a dual purpose during operation. It determines which one of a set of active cells deserves priority and therefore should be kept active. It also enables only certain cells to shift their contents during the repacking process. The priority line is discussed in more detail in Section 3.4.

3.1.3 Cell Operations

Each cell is capable of storing the pattern presented on the ISB, outputing its contents on the RSB, or matching its contents against the pattern on the ISB.

The input circuit is shown in Figure 3. When a cell's activity is set and the input signal lead (IS) is activated whatever pattern is carried on the input symbol bus is stored in that cell.

In the matching operation the contents of all cells are simultaneously matched against the pattern on the ISB. The matching circuit of cell i is shown in Figure 4. This operation is controlled by the match signal lead (MS). During matching the contents of each cell, say cell i, is compared with the contents carried on the ISB. If the comparison is successful, an internal signal is generated in cell i which sets the cell's activity if it was reset or maintains the set condition if it was

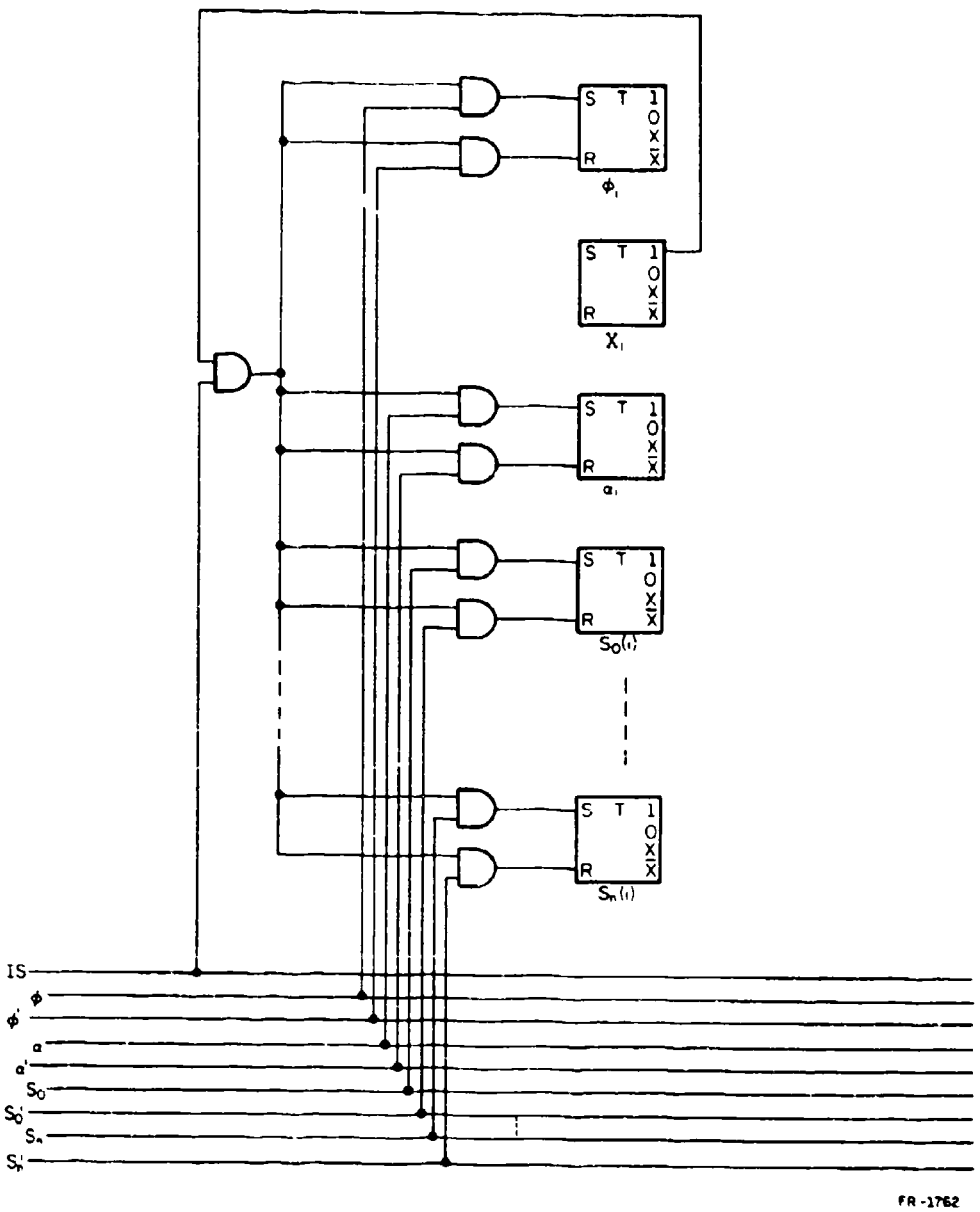

Figure 3

Input circuit of cell i

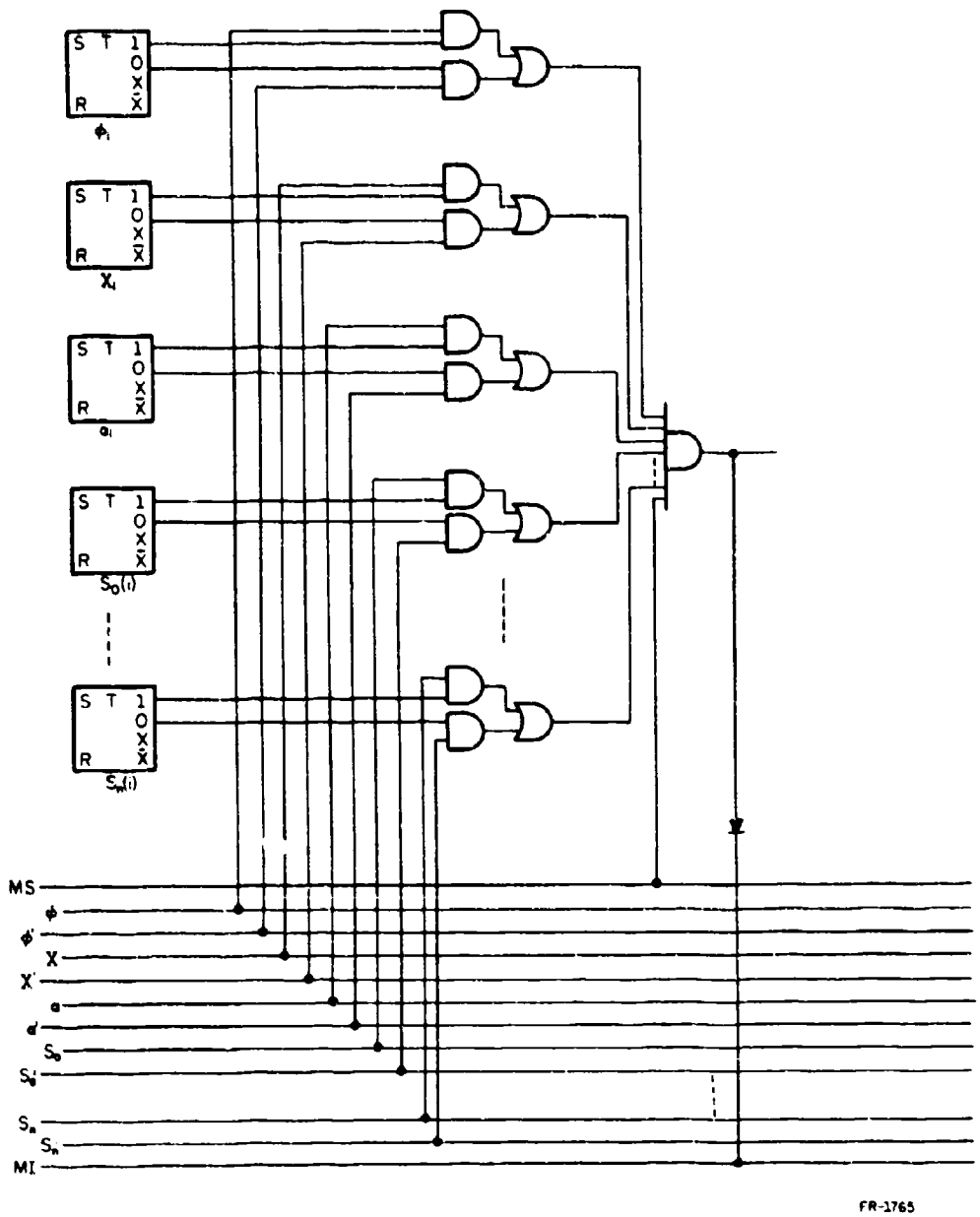

Figure 4

Matching circuit of cell i

originally set. If prior to a match operation a cell's activity was set and the match was unsuccessful in that cell the activity is reset.

The ISB lines consist of n+4 pairs of leads, $\emptyset$, $\emptyset'$, X, X', α, α', So, So', ... Sn, Sn'. This allows the indication of a $\emptyset$, 1 or a "don't care" condition for each of the n+4 bits of a cell. In this manner any subset of the set of n+4 bits of a cell may be used in the store or match operations. This requires that for matching, a "don't care" be specified by an $\emptyset$ on both leads.

The output circuit of cell i is shown in Figure 5. A cell may read out its contents onto the RSB lines if its activity is set and the read signal (RS) is activated. This information is transferred via the RSB to the control computer where it is either used for program direction or temporarily stored before being transferred to an output device (e.g. printer) or mass storage (e.g. disk, tape).

3.2 Command Lines

There are ten distinct command lines which are common to all of the cells of the associative memory presented above. Each of these lines controls a distinct operation which takes place simultaneously in all cells of the array.

Each of these ten command lines are now described below.

1) The "RS" (read signal) line, when activated causes the cell, which has its activity bit set, to output its contents onto the RSB lines.

2) The "IS" (input signal) lead enables all active cells to store the pattern which is present on the ISB lines.

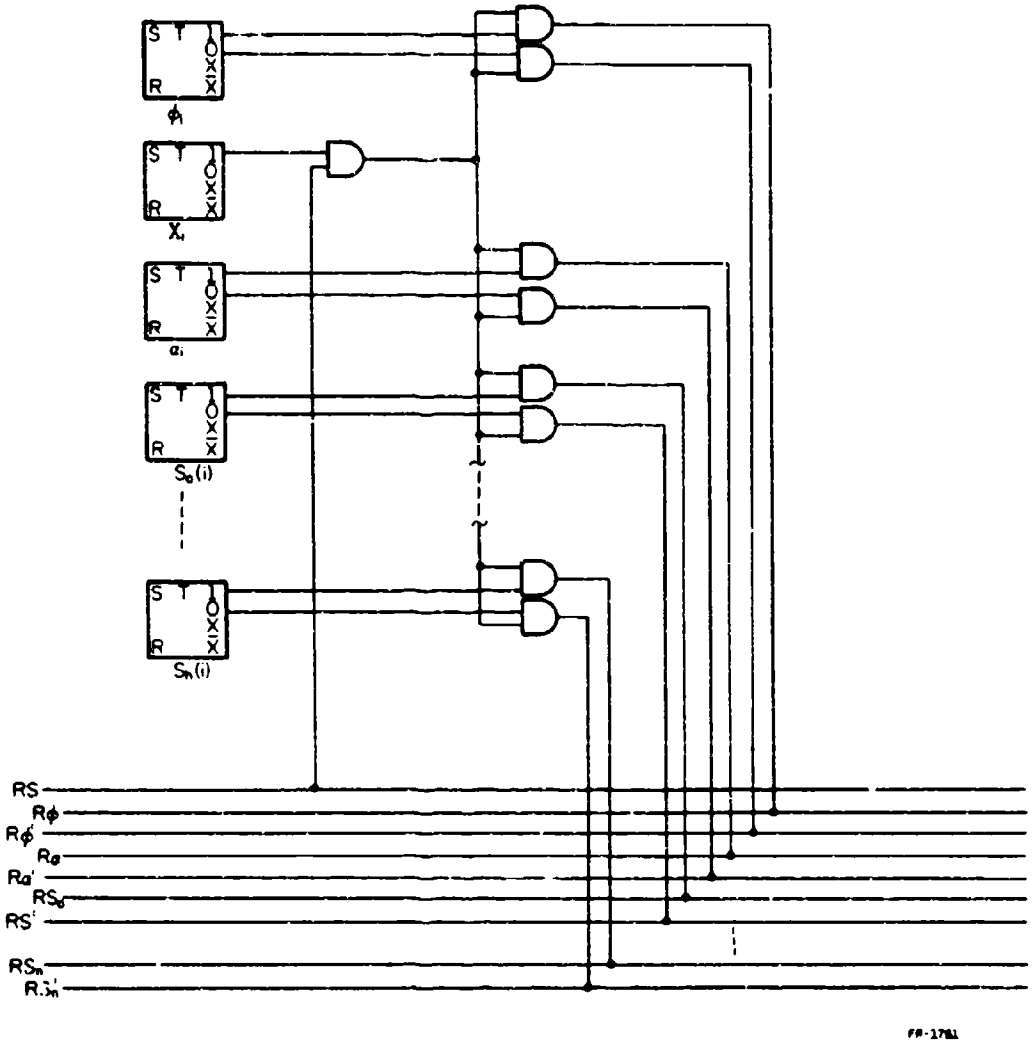

Figure 5

Output circuit of cell i

3) The "MS" (match signal) lead is activated only when matching is to take place.

4) Activating the "PR" (propagate activity) line causes all cells whose activity bits are set to set the activity of their neighbors to the right and then reset their own activity.

5) The "RA" (reset activity) lead causes the activities of all cells to be reset except during a match operation. Figure 6(a) shows the circuitry involved in a match and reset operation. Figure 6(b) shows the timing of the pulses on the RA and MS lines. If the MS line is not activated (no matching) the RA line may be used to directly reset all activities in the memory. If, during a match, a cell successfully matches its contents against the pattern on the ISB lines, the RA pulse is not allowed to reset that cell's activity bit.

6) The "SC" (score) line, when activated, causes all active cells in the memory to place a pulse on the routing line.

7) Activation of the "SE" (shift enable) line readies all cells to shift their contents to their neighbors to the left.

8) A pulse on the "CSL" (contents shift left) line causes the contents of all the cells enabled by the priority line to be shifted into the neighboring cell to the left.

9) The "PS" (priority select line), when activated, allows the activity bit of the leftmost active cell in the memory to remain set while all other active cells are reset.

10) The "LC" (priority line clear) line is used to eliminate excessive gate delay in the priority line during the repacking operation. This is explained in more detail in Section 3.4.

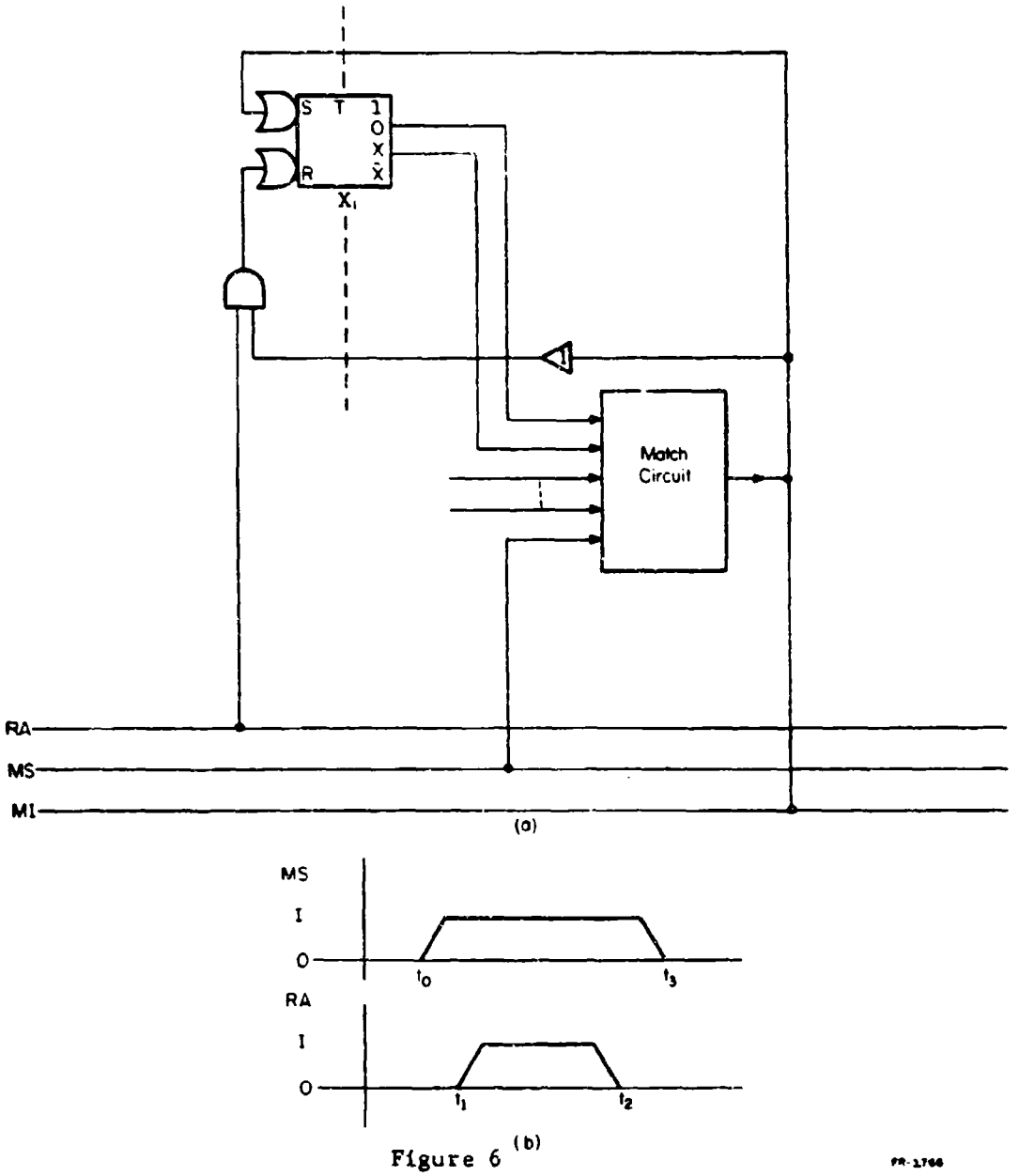

Figure 6

Resetting activity during match

a) circuitry involved

b) timing diagram

The last of the set of lines which are common to all the cells is the "MI" (match indication) line. The output of a cell's match circuit is connected to this line. In the control computer this line is fed to a threshold circuit which determines which of three conditions result from a matching operation. These are: a) no match is present; b) only one match is detected; or c) more than one match is present. This information is useful during processing.

3.3 The Routing Line

The routing line is a one-directional gated network which extends through each cell in the memory array. It is the presence of this line which allows a string of cells to act somewhat as an independent processing unit.

A four cell section of the routing line is displayed in Figure 7. The state of the α binary of each cell controls the section of the line related to that cell. If the α binary is set, as in cell i, that cell acts as a head cell and the pulses present on the section of the line to the right of this cell are allowed to pass into its accumulator. The condition on the "and" gate, associated with the head cell, in the routing line stops the pulses from travelling any farther to the left. If the α binary is reset, as in cells i+1, i+2, i+3, the pulses present on the line are blocked from entering the cell's register. In addition each cell in this state will place a pulse on the routing line if its activity bit is set and a SC command line is activated by the control computer. This enables a string of cells to store the results of a number of searches as a binary number in the symbol register of its head cell. This ability is

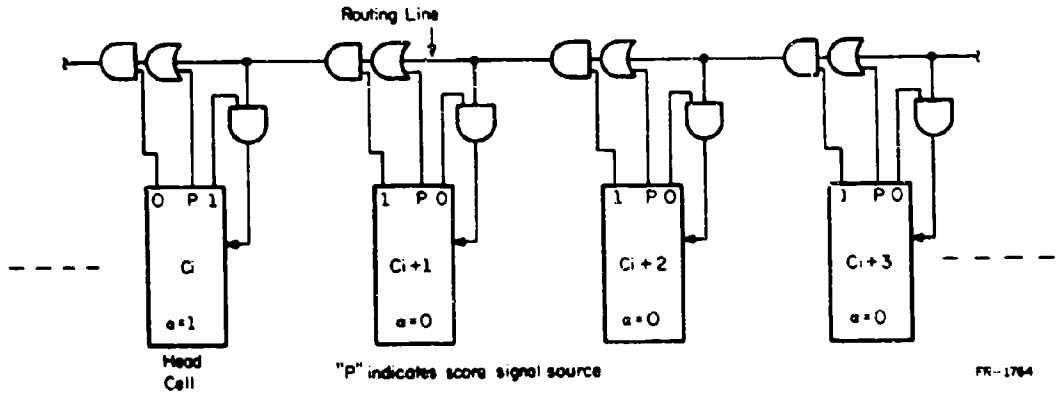

Figure 7

Section of routing line

necessary for the processing of certain types of searches which are important to information retrieval.

3.4 The Priority Line

A fast and efficient method of determining a priority among a set of active cells is of great importance in a processor of this type. In most cases the results of a search will yield a number of strings of information which are of interest for output. These strings will, of course, be located at arbitrary positions in the memory. They would be identified by having the activity bit of their head cells active. Since only one string of this set may be read out at a time, it would be necessary to be able to determine which string of this set should be read out first. It would also be advantageous to eliminate this string from further consideration after it had been read out; and be able to identify the rest of the strings in the set without performing the entire search procedure again. The speed of the priority system must be comparable to the commands issued on the control buses. If this were not so the priority system would present a bottleneck to the machine's operation.

Figure 8 illustrates the priority system of this processor. The priority line is a one-directional gated network which passes through the entire array of cells. In each cell the line passes through one AND gate and one OR gate. The PC input to the AND gate normally carries a "1" condition. The set output of each Ø binary is one of the inputs to the OR gate. This arrangement divides the entire priority line into two segments. The segment which is to the left of the first cell (the key

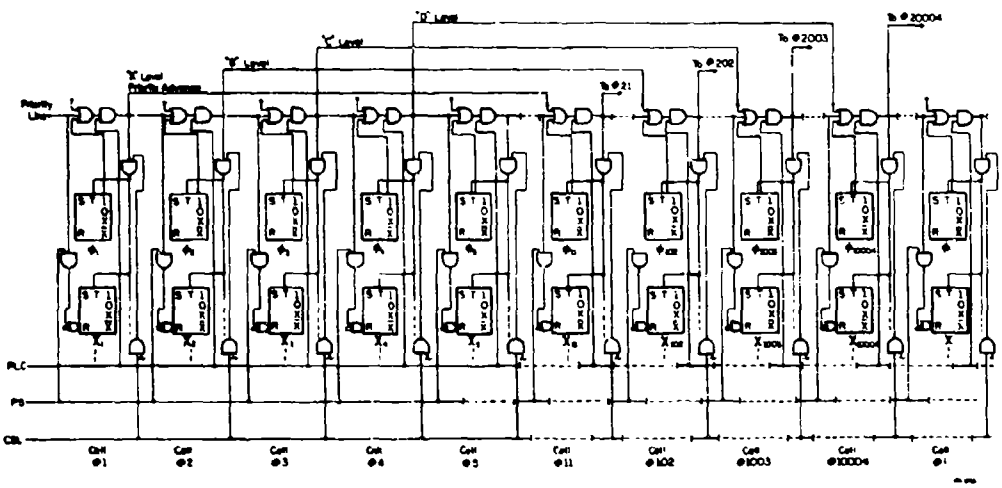

Figure 8

Priority line with priority advance

cell) with its Ø bit set to 1 (first from the left) is inactive. The segment to the right of this cell is active (carries a 1 condition). This condition allows the line to perform two important functions.

3.4.1 Priority of the Leftmost Active Cell

After a search procedure the symbol register of the head cells of the set of strings of interest contain a particular binary number. These head cells are identified by performing a single search. After this all the head cells of this set have their activity bits set. Next a store procedure sets the Ø bit in all active cells. The Boolean AND of the output of the priority line from cell i-1 and the PS line is gated into the reset of the activity bit of cell i, for all cells. Therefore when the PS line is activated only the leftmost active cell in the array retains its set activity. All other active cells are reset. The string to which this head cell belongs may now be read out on the RSB lines. To eliminate this string from further consideration, its head cell's symbol register and Ø bit are reset before the output procedure is begun. Now the new leftmost head cell of the set of strings can be found by repeating the procedure above. One restriction on this scheme is that the memory must be tightly packed before the priority system is used. This provides for the condition that the only cells in the data base which have the Ø bit set are the head cells belonging to the strings of interest.

3.4.2 Repacking Operation

In order to insure the full utilization of all the cells in the memory array, it is necessary to eliminate gaps consisting of empty cells

which may exist between strings of useful data. The repacking operation provides this service.

After the memory is initially loaded no gaps exist among the data strings. All of the empty cells (with Ø set) are in a contiguous group of cells located in the right most part of the array. A gap is introduced by deleting a particular string (setting Ø=1 in all cells of the string). To close this gap it is necessary to have all the cells to the right of the first empty cell, from the left, (key cell) shift their contents into their left neighbor as many times as there are empty cells in the gap. During the packing operation the priority line enables all the cells to the right of, and including, the key cell to receive the pulse sent along the CSL line, while all of the cells to the left of this cell are kept from shifting. Repacking is performed immediately after the deletion of a string of cells. The control computer counts the number of empty cells stored and then shifts the array this number of times.

3.4.3 Priority Advance and Priority Clear

To eliminate excessive gate delay in the priority line two additional features were included.

A "turn on" delay would result from setting the Ø bit in a cell which is to the left of the previous key cell. The delay would result from the activating of that portion of the line between these two cells. A "worst" case would be if no cell in the array had its Ø bit of cell 1. In this case the entire line would have to be activated. By including the external connections called "priority advance lines," shown in Figure 8, this turn on delay can be kept to a minimum. Four levels of priority

advance are shown in the illustration. Level "A" begins at the output of cell 1 and is repeated every tenth cell. Level "B" starts at the output of cell 2 and is repeated every hundredth cell thereafter. Levels "C" and "D" are repeated every one thousand and ten thousand cells respectively. For an array consisting of one hundred thousand cells, using this type of priority advance, the worst case turn on delay for the entire line would be less than ninety gate delays. More levels of priority advance could be included for both larger arrays and faster operation.

A "turn off" delay would be present as the result of closing a gap of empty cells in the information strings. When the gap is finally closed a new priority must be established with regard to the new leftmost cell with its $\emptyset$ bit set. This would require that the processor wait until the section of the priority line between the previous key cell and the new key cell is deactivated. If this section includes many cells then the resulting gate delay would be very large. This delay can be eliminated by the use of the PC line and the AND gates associated with the line.

The PC line normally carries a "1" condition. This allows the outputs of the AND gates to depend solely on the condition of the priority line inputs. After a gap has been closed, the control computer issues a "0" pulse along the PC line. This simultaneously turns off all the AND gates in the line and this in turn deactivates the entire priority line. After the PC line resumes its normal state of "1" a new priority has been established with respect to the new key cell. Thus the "turn off" time has been eliminated and the processor must wait only long enough for a new priority to be established.

Chapter 4

Data and Programming Formats

4.1 Data Format

All information is stored in the memory as strings of characters. Each character of a string is stored in a distinct cell. The largest individual blocks of information stored in the memory will be called "records." The identification of a particular subset of the set of all records stored in the memory will be the object of a particular search. Each record in turn is composed of an arbitrary number of smaller information blocks called parameters. A number of known parameters or parts of parameters will be used as the identifying criteria during a search. This is to say that certain parameters are known and it would be desirable to locate all records which contain these parameters. A parameter consists of an indefinite number of characters. Each character is stored, in coded form, in the symbol register of a separate cell. Therefore the character can be considered the primary piece of data which is stored in the memory. The number of distinct characters to be used during processing will determine the optimum size of a cell's symbol register.

The location of a particular record in the memory is completely arbitrary, since the cells do not possess addresses. Likewise, the order of parameters in a record is not of great importance. What is important is that all parameters of a record be located in contiguous groups of cells. Also the characters of a parameter must be stored in adjacent cells, with their order maintained. As an example if the parameter XYZ

is to be stored in the memory, the character X would occupy cell i, with Y in cell c_{i+1} and Z in cell c_{i+2}.

The first cell in a record is always a head cell. The special symbol γ always proceeds and follows the parameters of a record. As an example, the record containing the parameters XY, OW, PHD, PCC and AF would be stored in the following manner:

---αγXYγOWγPHDγPCCγAFγγ---.

4.2 Instruction Format

A set of instructions can now be defined which will facilitate the programming of the processor. Each of these instructions belongs in either one of two categories. The first contains those that do not directly affect the cell memory and those that do. The former are necessary to direct the flow of programming in the control computer. The latter consist of a timed sequence of one or more pulses sent along the command lines to the cells of the associative memory. These instructions are stored as a program in the control computer and are executed by the control computer.

4.2.1 Instructions

There are a total of twelve basic instructions used in programming this processor. These are listed below along with a description of the command lines involved and the function performed. Each instruction may consist of at least two parts, an instruction number and an instruction name. An instruction number is necessary only if a transfer is to be made

to that instruction during the execution of the program. Otherwise, it may be omitted. Additional parameters will be included in certain instructions. Their use is explained in each particular case. All instructions are executed in order except in the case of a program transfer.

The following describes the set of instructions needed:

1.) /NUMBER/MATCH/SYMBOL/(0),(1),(>1)/

This is the MATCH instruction. When executed it instructs all cells in the associative memory to compare their register contents with the pattern presented on the ISB lines. If a cell's contents match this pattern, its activity bit is set. If the match is unsuccessful the cell's activity either remains reset or is reset, depending on its status before the matching operation.

The first part of the instruction is its identifying number or name. If not used, this may be omitted. The second part is the instruction name. The third contains the symbol or pattern which is to be placed on the ISB for comparison. The pattern to be used is specified by indicating the status of each of the three 1 bit fields and the character which is required in a matching cell's symbol register in the following form: /Ø,X,α;CHARACTER/. If a "don't care" condition is desired on the Ø,X or α binaries that position is left blank. For example, to match against the pattern Ø=1, X=don't care, α=0 and CHARACTER=B the third part of the instruction would contain /1, , 0; B/. If a "don't care" is required on one of the bits of the CHARACTER field, then instead of a character, each bit position is indicated with the "don't care" positions being left blank. As an example the pattern Ø=1, X=1, α=1, S_0=don't

care, S_1=1, S_2=1, ..., S_{n-1}=1, S_n=don't care would be written as /1,1,1;1,1,1, ...,1, ,/.

The final part of the match instruction designates a three-way conditional transfer which depends on the outcome of the search. This is somewhat similar to an "if" statement in a Fortran program. The first set of parentheses contains the number of the instruction to be executed next if the control computer receives indication (that no cell matches the pattern) from the MI line. The second set indicates the next instruction in case of only one cell matching. The third holds the number of the next instruction for the occurence of a multiple match (e.g. /(N_1),(N_2),(N_3)/). If the parentheses are empty, the program proceeds to the next instruction in order. The ISB, MS, MI, and RA lines are used in the execution of this instruction:

2.)/NUMBER/STORE/SYMBOL/

This instruction stores the given symbol (/Ø, ,α;CHARACTER/) in all cells in the memory which have their activities set (X=1). The same symbol format is used as in the MATCH instruction, except that the X position is always left blank because it is not possible to store a condition in that bit. A "don't care" condition is indicated in the same manner. The ISB and IS lines are used in this instruction:

3.)/NUMBER/READ/

This instruction orders the single active cell in memory to place the contents of its registers on the RSB lines. The pattern on the RSB lines is then stored in a special buffer register located in the control

computer. This buffer always contains the last pattern read out from the memory. From the buffer the pattern may be transferred to move permanent storage such as the control computer's core or a peripheral device. The RSB and RS leads are employed in this operation.

4.)/NUMBER/BMATCH/SYMBOL/(0),(1)/

This instruction is similar to the MATCH instruction except that the SYMBOL field is matched against the contents of the buffer register instead of the memory cell contents. In this case there is either a match or no match. Therefore, it is only a two-way conditional transfer.

5.)/NUMBER/PROP/

This instruction orders all active cells (X=1) to set the activity bit of their right neighbor and then reset their own activity. The PR line is used in this instruction.

6.)/NUMBER/RESET/

This instruction orders all cells in the memory to reset their activity bits. The RA lead is used.

7.)/NUMBER/SCORE/( )/

This instruction causes all cells in the memory with activity bits set to place a number of pulses on the routing line. The number of pulses is indicated between the parentheses of the third field of the instruction. The SC line is used.

8.)/NUMBER/DELETE/

This is a special form of the STORE instruction. It is used when information is to be eliminated from the memory. When it is executed the symbol register and α binary of an active cell are reset while the $\emptyset$ bit is set. Therefore, the symbol stored is /1, ,0,0,0, ...,0/. During the deletion of a portion of data the control computer automatically keeps track of the number of times the DELETE instruction is used. This number is then used during the repacking of the memory contents. A repacking operation must always follow a deleting routine

9.)/NUMBER/PACK/

The execution of this instruction automatically repacks the cell memory after a string of data has been eliminated. The contents of the cells to the right of the "key cell" (Section 3.4) are shifted to the left a number of times equal to the number of times the DELETE instruction was used. The leads involved are SE, CSL, PS, and PLC.

10.)/NUMBER/PRIORITY/

This instruction resets the activity bit in all cells except for the leftmost active cell in the array. This is accomplished through the use of the priority line. A PRIORITY instruction must be preceded by a STORE instruction which sets the $\emptyset$ bit in all cells whose activity bits are set. The PS line is used.

11.)/NUMBER/GOTO/( )/

The GOTO instruction is an unconditional transfer to the instruction whose number is contained in the parentheses.

12.)/NUMBER/HALT/

The HALT instruction stops the program.

The twelve instructions listed above are sufficient to process the data which is stored in the associative memory. An additional number of instructions would be required to carry out functions which involve only the control computer and peripheral devices (i.e. printer, typewriter, tape and disk storage). These would be of the same nature as instructions used in conventional processors.

4.3 "Mass Load" and "Mass Unload"

The ability of a cell to shift its entire contents into the registers of its neighbor to the left allows the memory to perform two very useful operations.

The first is "mass" loading of the memory. Initially the entire memory is empty and the data base must be stored before processing is begun. In this state every cell in the memory has only one bit active, the $\emptyset$ bit. The last cell in the memory, cell m (rightmost cell), is capable of shifting a pattern into its registers directly from the control computer. In the mass load operation data is shifted into and down the array via cell m. The shifting is continued until cell 1 is occupied. At this point the loading is complete. Since the array must be shifted as many times as there are cells in the memory the mass load is only

desirable if the amount of data to be loaded is close to the capacity of the memory.

During normal operation new data is more conveniently loaded using the ISB lines. This process is explained in the following section.

The second operation is "mass unloading." It might be desirable to empty the memory but save the data which it contains. This can be accomplished by shifting the data out, symbol by symbol, through cell 1 into the control computer. For this operation the entire priority line would be activated by an input to the line at cell 1 from the control computer.

In the following section, programs are described and examples are given which demonstrate the power of this system when applied to problems of information retrieval.

Chapter 5

Programs and Examples

The program instructions defined in the previous chapter can now be used to present a few examples of the processor's operation. These will serve to illustrate the usefulness of this system's application to problems which are of interest in the area of information retrieval. The examples have been chosen to demonstrate the important qualities of this organization. The problems to be discussed are: storing a record, identifying a set of strings given a parameter, deleting a set of strings, and a threshold search. In each example an ordered set of instructions, or program, is included and explained.

A minor restriction on the data stored in the memory is that, before a program is executed, all empty cells in the memory must be in a contiguous group in the right most part of the array. This divides the cells in the memory into two continuous strings. The other in the left part of the memory contains cells storing information. This restriction is minor because the data is originally loaded in this form and repacking is performed after each deletion operation.

5.1 Single Parameter Search

In this first example the object is to identify the set of all records, stored in the cell memory, which contain a particular parameter. For the purpose of illustration the parameter used as the search criteria is ABC. This parameter is the known portion of data which is to be used

in an associative search to find the set of all records which contain ABC as a parameter.

The search is performed for the contiguous group of cells γABCγ. This eliminates from consideration the records with parameters which contain, as a part, the character string ABC, e.g. γBABCγ, γABCDγ.

The program of instructions for this search is listed in Table 1. The first instruction (NUMBER=10) prepares the memory for the search by resetting all previously set activity bits. Instruction number 11 sets the activity bit in all cells which contain the character γ. After the execution of instruction number 19 the only cells in the memory whose activity bits are set are those which contain the character γ and are preceded, in order, by cells containing C, B, A and γ; i.e. cell i contains γ with activity bit set, cell i-1 contains C, cell i-2 contains B, cell i-3 contains A and cell i-4 contains γ. The execution of instruction 20 orders all active cells to place a pulse on the routing line. After this the head cells of all records, of which the parameter γABCγ is a part, contain a binary count of one in their symbol registers. This identifies the set of records sought. Instruction 21 resets all activity bits in the memory and instruction 22 stops the program.

5.2 Deletion of a Set of Records

The program listed in Table 2 will delete from the memory all the records which were identified in the previous example. These records are distinguished from all other records in the memory by the fact that their head cells' symbol registers contain a count of binary "1" (so bit is active).

Table 1

Program for finding the set of all
records which contain ¢ABC¢

```
/10/RESET/
/11/MATCH/0,0,0;¢/(22),(  ),(  )/
1/12/PROP /
/13/MATCH/0,1,0;A/(22),(  ),(  )/
/14/PROP/
/15/MATCH/0,1,0;B/(22),(  ),(  )/
/16/PROP /
/17/MATCH/0,1,0;C/(22),(  ),(  )/
/18/PROP/
/19/MATCH/0,1,0;¢/(22),(  ),(  )/
/20/SCORE/(1)     /
/21/RESET/
/22/HALT /
```

Table 2

Program for deleting all records
tagged in the example of Section 4.1

```
/50/MATCH   /0,0,1;1,0,...,0/(56),(51),(51)/
/51/STORE   /1, , ;          /
/52/PRIORITY/
/53/DELET   /
/54/PROP    /
/55/READ    /
/56/BMATCH  /0,1,1;          /(57),(60)/
/57/DELETE  /
/58/PROP    /
/59/GOTO    /(55)            /
/60/RESET   /
/61/MATCH   /1,0,1;          /(65),(62),(62)/
/62/STORE   /0, , ;          /
/63/PACK    /
/64/GOTO    /(50)            /
/65/RESET   /
/66/HALT    /
```

Instruction 50 of this program sets the activity bit in the head cells of these records. The next instruction prepares for a priority operation by storing a "1" in the ∅ bit of these active cells. After the execution of instruction 52 only the leftmost active cell (head cell in this case) in the memory retains its set activity, all others are reset. Instruction 53 deletes this head cell. Instructions 55 through 59 proceed to delete characters in the string until the next head cell is encountered. This signifies the end of this record. Next the head cells of the remaining records of the set are found and the ∅ bits are reset (instructions 60-62). This allows the repacking operation to proceed correctly (instruction 63). Due to instruction 64 the process continues to repeat itself until all records of the set are eliminated from the memory. This program both deletes the set of records and repacks the memory.

5.3 Adding a Record

During the operation of the processor, it may be necessary to periodically add information to the data already stored in the cell memory. This is accomplished by adding the record to the end of the string of cells containing information. First, the leftmost e pty cell is found and then characters are stored, one by one, in succeeding empty cells until the complete record is entered. The control computer automatically keeps track of the number of empty cells in the array, and therefore can determine if there is room enough in the memory for a new record.

Table 3 lists the program for entering the record $\alpha\gamma AB\gamma BC\gamma$. Instructions 100 and 101 locate and set the activity in the leftmost empty cell. The remainder of the program stores the characters of the new

Table 3

Instructions for storing the string $\alpha\gamma AB\gamma BC\gamma$

```
/100/MATCH   /1,0,0;0,....,0/(117),(117),(   )/
/101/PRIORITY/
/102/STORE   /0, ,1;0,....,0/
/103/PROP    /
/104/STORE   /0, ,0;γ       /
/105/PROP    /
/106/STORE   /0, ,0;A       /
/107/PROP    /
/108/STORE   /0, ,0;B       /
/109/PROP    /
/110/STORE   /0, ,0;γ       /
/111/PROP    /
/112/STORE   /0, ,0;B       /
/113/PROP    /
/114/STORE   /0, ,0;C       /
/115/PROP    /
/116/STORE   /0, ,0;γ       /
/117/HALT    /
```

record. This is accomplished by successively storing in the empty active cell and then propagating the activity to the next empty cell. Figure 9 shows the portion of the array involved before, during and after the storing procedure. Figure 9a shows the condition of the memory before the program is started. All cells to the left of the cell containing γ hold information, while all the cells to the right of this cell are empty. Figure 9b shows the condition of the memory after instruction 109 has been executed. In Figure 9c the record has been stored and a "new" leftmost empty cell is indicated.

5.4 Threshold Search

One important advantage of this processor is its ability to store the results of successive searches. This can clearly be demonstrated in a threshold search.

In this example the data base stored in the cell memory will consist of the three records αγADγDEFγBADγ, αγADγCADγ and αγDEFγBACγ. These are stored in the manner shown in Figure 10a. The problem is to identify and read out all records which contain two or more of the parameters AD, DEF and BAC. The program listed in Table 4a searches for the given parameters and tags the head cells of the records to which the various parameters belong. The parameters are located in the same way as in Section 5.1.

Instructions 1 to 8 of Table 4a locate all occurrences of the parameter γABγ. Instruction 9 records a binary 1 in the symbol registers of the head cells whose records contain this parameter. At this point the condition of the memory is shown in Figure 10b. The binary count of

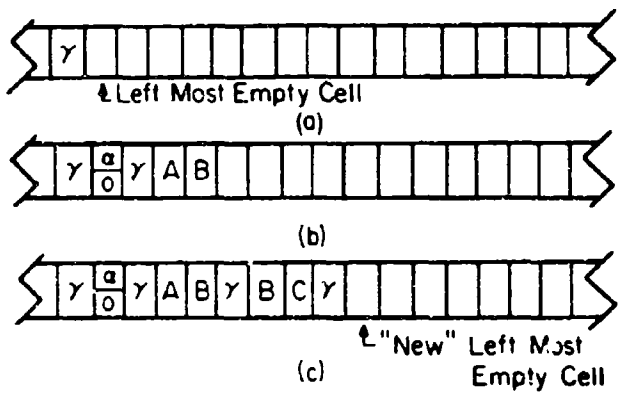

Figure 9

Contents of cell memory

a) before

b) during

c) after

the storage of record αγABγBCγ

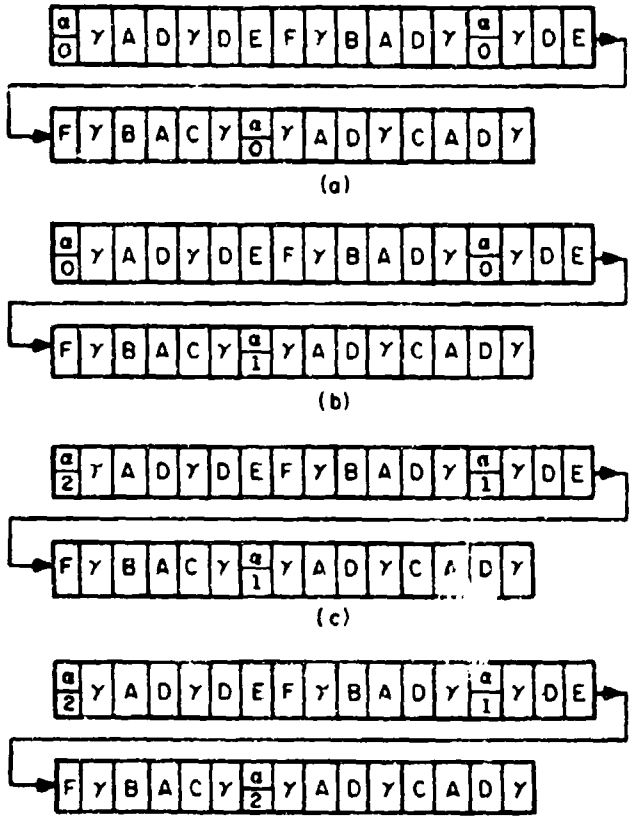

Figure 10

Threshold search FR-1760

a) condition of memory before search

b) after searching for γABγ

c) after searching for γDEFγ

d) after searching for γBACγ

Table 4a

Threshold search: Finding Records

```
/ 1/RESET//
/ 2/MATCH/0,0,0;γ/(32),(  ),(  )/
/ 3/PROP /
/ 4/MATCH/0,1,0;A/(10),(  ),(  )/
/ 5/PROP /
/ 6/MATCH/0,1,0;B/(10),(  ),(  )/
/ 7/PROP /
/ 8/MATCH/0,1,0;γ/(10),(  ),(  )/
/ 9/SCORE/(1)     /
/10/RESET/
/11/MATCH/0,0,0;γ/(32),(  ),(  )/
/12/PROP /
/13/MATCH/0,1,0;D/(21),(  ),(  )/
/14/PROP /
/15/MATCH/0,10;E/(21),(  ),(  )/
/16/PROP /
/17/MATCH/0,1,0;F/(21),(  ),(  )/
/18/PROP /
/19/MATCH/0,1,0;γ/(21),(  ),(  )/
/20/SCORE/(1)     /
```

Table 4a Continued

```
/21/RESET/
/22/MATCH/0,0,0;γ/(32),(  ),(  )/
/23/PROP /
/24/MATCH/0,1,0;B/(32),(  ),(  )/
/25/PROP /
/26/MATCH/0,1,0;A/(32),(  ),(  )/
/27/PROP /
/28/MATCH/0,1,0;C/(32),(  ),(  )/
/29/PROP /
/30/MATCH/0,1,0;γ/(  ),(  ),(  )/
/31/SCORE/(1)     /
/32/RESET/
/33/HALT/
```

of a record is given in decimal form in the lower part of its head cell. Next instructions 10 through 20 perform the same operations for the parameter γDEFγ. After this, the condition of the memory is as shown in Figure 10c. With instructions 21 through 32 the search is again repeated for the parameter γBACγ. Figure 10d shows the final condition of the memory. At this point two head cells contain a count of 2 while the third contains a count of 1. The set of all records which contain two or more of the three given parameters can now be found by searching for a head cell with a particular count in its symbol register.

The program given in Table 4b performs the task of locating and reading out these records. Instructions 34 throuth 42 first read out all records which contain all three given parameters. Then the records which contain only two of the three parameters are outputted by instructions 43 to 52. As a record is read out it is eliminated from further consideration by instructions 38 and 47 which reset the symbol register of its head cell.

A weighted threshold search could be performed in the same manner. The difference would be in the number of SCORE pulses associated with a particular parameter. This number would vary with respect to the relative importance of the parameter.

Table 4b

Threshold search: outputting records

```
/34/RESET     /
/35/MATCH     /,0,1;1,1,0,...,0,/(43),(36),(36)/
/36/STORE     /,1, , ;          /
/37/PRIORITY  /
/38/STORE     /0, , ;0,0,0,...,0/
/39/PROP      /
/40/READ      /
/41/BMATCH    / ,1,1;           /(42),(34)      /
/42/GOTO      /(39)             /
/43/RESET     /
/44/MATCH     / ,0,1;0,1,0,...,0/(52),(45),(45)/
/45/STORE     /1, , ;           /
/46/PRIORITY  /
/47/STORE     /0, , ;0,0,...,0/
/48/PROP      /
/49/READ      /
/50/BMATCH    / ,1,1;           /(51),(43)      /
/51/GOTO      /(48)             /
/52/RESET     /
/53/HALT      /
```

References

[1] Lee, C. Y. and M. C. Paull, "A Content Addressable Distributed Logic Memory with Application to Information Retrieval," Proceedings of the I.E.E.E., Vol. 51, pp. 925-932, June 1963.

[2] Lee, C. Y., "Intercommunicating Cells--Basis for a Distributed Logic Computer," Proc. Fall Joint Computer Conf., Vol. 22, pp. 130-136, December 1962.

[3] Gaines, R. S. and C. Y. Lee, "An Improved Cell Memory," I.E.E.E. Trans. on Electronic Computers, Vol. C-17, No. 1, pp. 10-17, January 1968.

[4] Sturman, J. N., "An Iteratively Structured General Purpose Digital Computer," I.E.E.E. Trans. on Electronic Computers, Vol. EC-14, No. 1, pp. 2-9, January 1968.

[5] Salton, G., "Progress in Automatic Information Retrieval," I.E.E.E. Spectrum, pp. 90-103, August 1965.

[6] Hayes, J. P., "A Content Addressable Memory With Applications to Machine Translation," University of Illinois, Digital Computer Laboratory Report No. 227, June 1, 1967.

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
University of Illinois Coordinated Science Laboratory Urbana, Illinois	Unclassified 2b. GROUP

3 REPORT TITLE

AN ASSOCIATIVE PROCESSOR FOR INFORMATION RETRIEVAL

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

5 AUTHOR(S) (First name, middle initial, last name)

Kisylia, Andrew Philip

6 REPORT DATE	7a. TOTAL NO OF PAGES	7b. NO OF REFS
August, 1968	50	6

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAB-07-67-C-0199; also in part OE C-1-7-071213-4557.	R-390
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10 DISTRIBUTION STATEMENT

Distribution of this report is unlimited.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
	Joint Services Electronics Program thru U.S. Army Electronics Command Ft. Monmouth, New Jersey 07703

13 ABSTRACT

NONE

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Content addressable						
Information retrieval						
Documanet retrieval						
Associative memory						
Cell memory						